University-Enterprises Cooperation
In Game Industry In **Ukraine**

561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

ОСОБЛИВОСТІ ВИКОРИСТАННЯ ФРЕЙМВОРКУ <<SPRITE KIT>> ДЛЯ ПРОГРАМУВАННЯ ІГОР ПІД IOS

Co-funded by the
Erasmus+ Programme
of the European Union

Структура Sprite Kit Framework

SKView и SKScene

У Sprite Kit анімація і рендеринг відбуваються всередині SKView.

Початково необхідно розмістити свій SKView в контролері і потім заповнити його сценами. Сцени представлені в Sprite Kit об'єктами SKScene. Як правило, сцена містить спрайт і інші об'єкти для відображення. В один момент часу, SKView може показувати тільки одну сцену, поки сцена є видимою її анімація і логіка виконуються автоматично кадр за кадром. Можна легко використовувати один SKView для показу різних сцен і переходів між ними. Таким чином, при розробці гри вам знадобляться один або кілька підкласів SKScene. Зазвичай різні підкласи відповідають за головне меню, за саму гру, за екран завершення гри і ін.

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

SKNode

Клас SKScene є нащадком класу SKNode. При використанні Sprite Kit, вузли є основними будівельними блоками для всього контенту, в тому числі і SKScene, яка виступає в якості кореневого вузла для дерева об'єктів-вузлів. SKScene і його нащадки визначають, який контент обробляти і як його візуалізувати. Також батьківський клас задає відношення до інших властивостей вузла і його нащадків. Наприклад, коли вузол обертається, всі його нащадки також обертаються. Таким чином, можна побудувати складний образ, використовуючи дерево вузлів, а потім регулюючи властивості самого верхнього вузла застосувати до образу різні ефекти, наприклад: поворот, масштабування, змішати все зображення і т.д.

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

Текстури

Це зображення, які використовуються для відтворення спрайтів. Вони використовуються для покращення деталізації об’єктів, і створюються шляхом завантаження файлів зображень. Однак, Sprite Kit може створювати текстури в реальному часі з інших джерел, наприклад з Core Graphics. Також Sprite Kit може перетворити ваш набір вузлів з ефектами в текстуру. Sprite Kit автоматично управляє текстурами. Однак якщо у вашій грі використовується велика кількість зображень, то ручне управління може зменшити використання ресурсів процесора.

Атлас текстур - це група пов'язаних текстур, які використовуються разом у вашій грі. Наприклад, можна використовувати атлас текстур для зберігання всіх текстур, що відносяться до анімації одного персонажа. Атласи також використовуються для зменшення навантаження на процесор

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

SKAction

Кожна дія на сцені є об'єктом класу SKAction. У відповідь на команду виконання дії під час того, коли сцена циклічно обробляє кадри анімації, дія виконується. Деякі дії можуть бути завершені в одному кадрі анімації, в той час як інші дії можуть виконуватися протягом декількох кадрів.

Найбільш поширене застосування SKAction - анімація вузлів гри. Наприклад, за допомогою SKAction можна перемістити вузол, змінити його масштаб або повернути, зробити його прозорим.

Додатково, SKAction можуть також змінити дерево вузлів, відтворювати звуки, або виконувати власний код.

Можна комбінувати SKAction для створення більш складних ефектів. Можна також створювати групи SKAction, які будуть виконуватися одночасно або послідовно.

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

SKPhysicsBody, SKPhysicsJoint і SKPhysicsWorld

Нерідко виникає необхідність у взаємодії вузлів один з одним. Вузли можуть стикатися один з одним, збільшувати швидкість після зіткнення або падати в просторі під дією сили тяжіння.

Для цього, для кожного вузла необхідно створити «фізичне тіло» SKPhysicsBody і задати йому необхідні фізичні характеристики (форма, розмір, маса і т.д.).

Коли SKPhysicsBody прикріплені до сцени, сцена починає імітувати фізичні закони. Деякі сили, такі як тертя і гравітація, застосовуються в сцені автоматично. Також можна створити власні сили і правила взаємодії вузлів на сцені. При відтворенні сцени обчислюються прискорення і швидкість кожного вузла, після чого позиція і орієнтація відповідних вузлів змінюється візуально. При цьому можна точно контролювати взаємодію вузлів на сцені, використовуючи callback. Це дозволяє задавати ігрову логіку - наприклад, можна викликати callback закінчення гри, коли один вузол стикається з іншим.

Для того щоб визначити глобальні фізичні характеристики всієї сцени, використовується SKPhysicsWorld. Наприклад ви можете задати гравітацію для всієї сцени. Для моделювання фізичної поведінки декількох вузлів, використовується SKPhysicsJoint.

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

Загальний огляд Objective-C

Для позначення об'єктів використовується спеціальний тип id.

Змінна типу id фактично є покажчиком на довільний об'єкт.

Для позначення нульового покажчика на об'єкт використовується константа nil.

Замість id можна використовувати і більш звичне позначення з явним зазначенням класу.

Аналогом виклику методів в C++ (наприклад: objectName->MethodName) в Objective-C є «передача повідомлень».
Для передачі повідомлень використовується наступний синтаксис:

[receiver message];

де receiver - покажчик на об'єкт ; message - ім'я методу

Всі директиви, які використовуються Objective-C починаються з символу “@”

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

Класи в Objective-C

Оголошення класу в файлі з розширенням .h :

```
@interface ім'я_класу: super_Class
{   оголошення змінних; (Мають зону видимості private)
}
   оголошення методів;
@end
```

Реалізація класу у файлі з розширенням .m :

```
#import "ім’я_класу.h"
@implementation ім'я_класу
   реалізація методів;
@end
```

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

Файл first.h

```
#import <Cocoa/Cocoa.h>
@interface first : NSObject
{
        int a;
        int b;
}
- (id)init;
- (void)setA :(int)A andB :(int)B;
- (int)sum;
- (long)mul;
@end
```

Файл first.m

```
#import "first.h"
@implementation first
- (id) init  {   self = [super init];
   if (self) {     // ініціалізація класу
   }
   return self;
}
- (void) setA: (int) A andB: (int) B {
   a = A;   b = B;
}
- (int) sum {   return a + b; }
- (long) mul {   return a * b; }
@end
```

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

Властивості

Починаючи з Objective C 2.0 з'явилося поняття «властивість» якій відповідає директива @property, яка вказує, що дана змінна класу має метод для читання і запису в неї. Дана директива оголошується в файлі оголошенні класу.

Супутня директиві @property, директива, яка говорить компілятору, про те, що необхідно створити методи для доступу до змінної - носить найменування @synthesize і оголошується в файлі реалізації класу. Використання директиви @synthesize однак не є обов'язковим.

Синтаксис оголошення властивості:

@property (атрибути, через кому) тип ім'я;

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561726-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

ПРИКЛАД ПРОГРАМУВАННЯ ПРОСТОЇ ГРИ

Розглянемо приклад використання Sprite Kit та Objective-C, для написання простої тетрісоподібної гри. Від класичного Тетрісу така гра буде відрізнятись тим, що гравець може набирати бали за допомогою складання слів із букв, які падають випадково генеруючись, в кожній із 4х комірок тетрісових фігур. Причому, саме таким способом можна заробити більше очок, якщо складати якомога довші слова, оскільки кількість балів в такому разі розраховується як:

$n = m^3$, де n – кількість очок, а m – кількість букв у складеному слові.

Якщо гравець грає за правилами класичного Тетрісу, тоді за кожний повністю заповнений рядок, йому нараховується 10 очок.

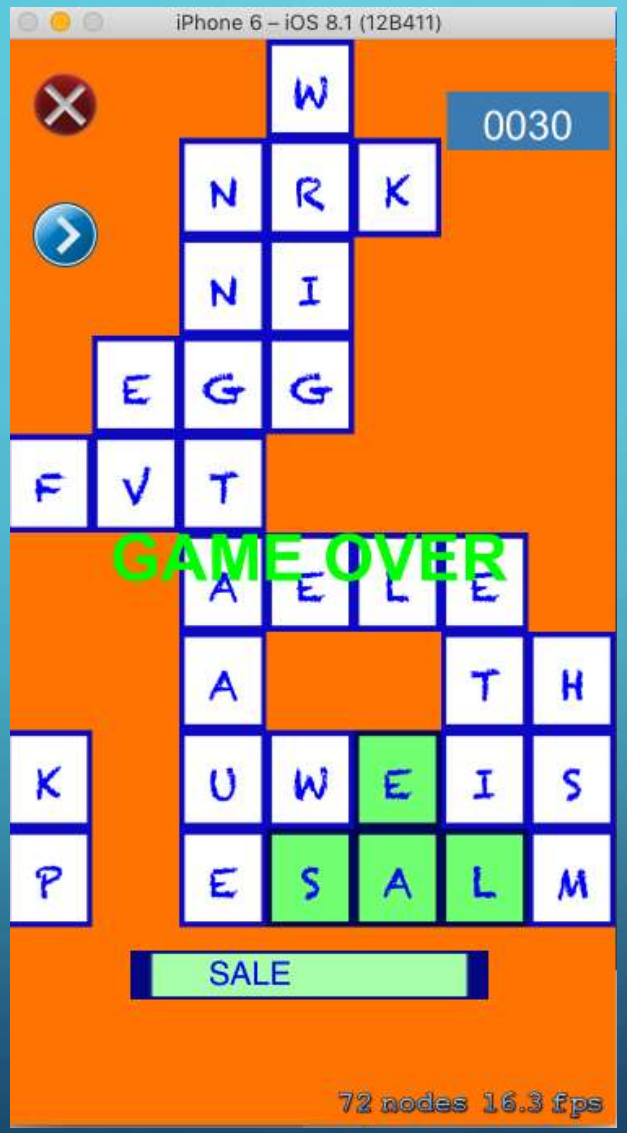
Таким чином, складання слів з кількістю букв >3 є стратегічно більш вигідним у грі.

Після складання слова, букви і їх клітинки зникають, а стоячі вище випадково опадають, що додатково ускладнює процес проходження гри.

Розглянемо приклад використання Sprite Kit та Objective-C, для написання простої тетрісоподібної гри. Від класичного Тетрісу така гра буде відрізнятись тим, що гравець може набирати бали за допомогою складання слів із букв, які падають випадково генеруючись, в кожній із 4х комірок тетрісових фігур. Причому, саме таким способом можна заробити більше очок, якщо складати якомога довші слова, оскільки кількість балів в такому разі розраховується як:
$n = m^3$, де n – кількість очок, а m – кількість букв у складеному слові.
Якщо гравець грає за правилами класичного Тетрісу, тоді за кожний повністю заповнений рядок, йому нараховується 10 очок.
Таким чином, складання слів з кількістю букв >3 є стратегічно більш вигідним у грі.
Після складання слова, букви і їх клітинки зникають, а стоячі вище випадково опадають, що додатково ускладнює процес проходження гри.

Co-funded by the Erasmus+ Programme of the European Union

ПРИКЛАД РОБОЧОГО ПОЛЯ ПРОТОТИПУ ГРИ



GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561726-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

Ініціалізація PlayScene

```
@implementation PlayScene
{ enum SceneState { CREATED=0, INITIALIZED, PAUSED, ACTIVE };
int state;
CGPoint prevloc;}
-(void)addControls {
self.backgroundColor = [SKColor orangeColor];
 //--------------------- "OPTIONS" scene button ---------------
 SKSpriteNode *button = [SKSpriteNode spriteNodeWithImageNamed:@"nextButton.png"];
 button.xScale = button.yScale = 0.4;        double x= self.w / 10;
 double y=CGRectGetMaxY(self.frame) - 3 * button.size.height;
 button.position = CGPointMake(x, y);    button.name = @"nextButton";    [self addChild:button];
  //----------- add footer ---------
SKSpriteNode* footer = [SKSpriteNode spriteNodeWithImageNamed:@"LetterBox.png"];
 footer.name = @"footer";    footer.size = CGSizeMake(_w * 0.8, _h/20);.....
```

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

Запуск фонової музики

```
-(void)didMoveToView:(SKView *)view {
//----- BackGround music----------
NSURL* soundFileURL = [[NSBundle mainBundle] URLForResource:@"mainMusic“ withExtension:@"mp3"];
NSError* error=nil;
if(!self.audioPlayer.isPlaying)    {
            self.audioPlayer= [[AVAudioPlayer alloc] initWithContentsOfURL:soundFileURL  error:&error];
            self.audioPlayer.volume=0.0;        self.audioPlayer.numberOfLoops=-1;
            [self.audioPlayer prepareToPlay];        [self.audioPlayer play];
}
switch(state)    {
   case CREATED:          [self addControls];          [_logic startGame:_level];          break;
                }
}
```

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

Обробка натискань по екрану

```
-(void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event {
   /* Called when a touch begins */
UITouch *touch = [touches anyObject];  prevloc = [touch locationInNode:self];
SKNode *node = [self nodeAtPoint:prevloc];
_blockPosition = node.position;
if ([node.name isEqualToString:@"nextButton"])
 { OptionsScene *optionsScene = [[OptionsScene alloc] initWithSize:self.size];          optionsScene.returnScene=self;
optionsScene.audioPlayer=self.audioPlayer;                    optionsScene.audioFxPlayer=self.audioPlayerFx; optionsScene.SFXVolume
= &_SFXvolume;
                optionsScene.MainVolume = &_MainVolume;
                SKTransition *transition = [SKTransition flipVerticalWithDuration:0.5];
                optionsScene.scaleMode = SKSceneScaleModeAspectFill;
                [self.view presentScene:optionsScene transition:transition];
     return;   }
else
 if ([node.name isEqualToString:@"rotateButton"]) {
              [_logic rotateBlock];           }
 else ......
```

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

Обробка часових відліків

```
-(void)update:(NSTimeInterval)currentTime{
 if(state==ACTIVE)    {
    [_logic update:currentTime];
    if(_logic.state == ROUNDFAULT || _logic.state == ROUNDDONE)
     {  state = LEVELSELECT;                    RWGameData* gameData = [RWGameData loadInstance];
          gameData.score = [_logic getScore];     gameData.maxRoundOpen = [_logic getMaxRoundNum];
          if(gameData.score > gameData.highScore)          {   gameData.highScore = gameData.score;          }
          if(_logic.state == ROUNDDONE)          {  [self gameOverON:@"LEVEL COMPLETE!"];
             if([_logic getRoundNum] == [_logic getMaxRoundNum])   { gameData.maxRoundOpen++;
                [_logic setMaxRoundNum: gameData.maxRoundOpen];   [_logic setRoundNum: gameData.round];
           }
              [_logic setRoundNum:gameData.maxRoundOpen];
    }      else          {              [self gameOverON:@"GAME OVER"];          }
    [gameData save];
    SKAction *delay = [SKAction waitForDuration:10];
        [self runAction:delay completion:^    {   [self gameOverOff];    [self runHighScoreScene]; }];
}}}
```

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

Структура модуля Logic. Внутрішні змінні

```
@interface Logic : NSObject
{
        int h;    int w;    double fallTime;    double gameMapSizeXpx;
        double gameMapSizeYpx;        double gameMapOffsetX;
        double gameMapOffsetY;
        GBlock* gameArr[arrXn][arrYn];
        CGSize blockSize;
        TetrisBlock* tBlock;          SKScene* scene;    GBlock* tempBlock;
        GBlock* wordArrLabel[WORD_LABEL_SIZE];
        int wordLen;    NSMutableSet* dictionary;    double timeStartPnt;
        double operationTime;        int gameScore;    int roundNum;
        int currentMaxRoundNum;    int currDestScore;
        double startTime, lastTime;    BOOL    isTimeGenMethod;    double timeInterval;
}
```

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561726-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

Структура модуля Logic. Методи

@property(nonatomic, strong) AVAudioPlayer *audioPlayer;

@property(nonatomic, strong) AVAudioPlayer *audioPlayerFx;

- (void) fadePlayer:(AVAudioPlayer*)player fromVolume:(float)startVolume toVolume:(float)endVolume overTime:(float)time;

@property int state; @property double* SFXvolume;@property double* MAINvolume;

@property (weak, nonatomic) ScoreForm* scoreLabel;

@property NSString* wordLabelText;@property UIColor* wordLabelColor;

-(id) initWithScene : (SKScene*)Scene andHeight: (float)height;

-(CFTimeInterval)currentTime; -(pos) CoordToPosInArr: (CGPoint) coord;-(CGPoint) PosInArrToCoord:

-(pos) position;-(CGSize) getBlockSize;-(int) getRoundNum;-(void) setRoundNum:

-(int) getMaxRoundNum; -(void) setMaxRoundNum: (int)n;-(void) calcDestScore;

-(long int) getDestScore;-(long) getScore;-(void) rotateBlock;-(void) showBanner;-(void) moveRightBlock;

-(void) moveLeftBlock;-(void) fastFall;-(void) moveDownBlock;-(BOOL) checkMoveFig: -(BlockAction) action;

-(BOOL) checkRotateFig; -(BOOL) checkSubNodeInTetrisBlock:(NSString*)nodeName; -(int) checkFilledLines;

-(void) fallBlock: -(void) removeFreeLines;-(void) createBlock; -(void) putTetrisBlockToGArr;-(void) artisticFall;

-(BOOL) pressWordButton; -(void) addLetterToWordLabel: (SKNode*) node; -(void) removeWordsBlocks;

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

Вилучення букв і їх блоків які утворили слово

```
-(void) removeWordsBlocks
{    int i=0;    int x=0,y=0;
 while(wordArrLabel[i]!=nil)    {
     [wordArrLabel[i] removeFromParent];
     for(x=leftX; x<rightX; x++)
          for(y=bottomY; y<=topY; y++)
          if(wordArrLabel[i]==gameArr[x][y]) goto L1;
L1:     wordArrLabel[i]=nil;
          for ( ; y <=topY; y++)       {
          gameArr[x][y] = gameArr[x][y+1];
          double fallTime_ = 0.5 + 0.5 * rand()/RAND_MAX;
          SKAction* down=[SKAction moveByX: 0.0 y: -tBlock.blockHeight duration:fallTime_];
          [ gameArr[x][y] runAction:down];
          }
 }}
```

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

Перевірка можливості руху фігури

```
-(BOOL) checkMoveFig: (BlockAction) action  {
pos posInGArr[4];
for(int i=0; i<4; i++){ posInGArr[i]=[tBlock getPositionInGameArr:i]; }
switch (action) {
            case RIGHT_A: { for (int i=0; i<4; i++) {
            If (gameArr[posInGArr[i].x+1][posInGArr[i].y]!=nil)  return NO;}}        break;
            case LEFT_A:    { for (int i=0; i<4; i++) {
            if (gameArr[posInGArr[i].x-1][posInGArr[i].y]!=nil) return NO;}}        break;                case
DOWN_A: { for (int i=0; i<4; i++) {
            if (gameArr[posInGArr[i].x][posInGArr[i].y - 1]!=nil)   return NO;}}     break;                case
ROTATION_A:  case STOP_A:   {return NO; }}
return YES;
}
```

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

Приклад побудови Т-подібної фігури

```
... case TFIG: // Create new T-type figure
 { y0= y0b - subBlockSize.height;
   for(int i = 0; i < 3; i++) {
                pos = CGPointMake( x0 + ( i * subBlockSize.width), y0);
                GBlock* block=[GBlock newBlockWithPos:pos withSize:subBlockSize parent:tBlock];
                tBlock->figArr[i] = block;  tBlock->figArr[i].hidden = NO;
                tBlock->blockPositionInGameArr[i].x = 4 + i;
                tBlock->blockPositionInGameArr[i].y = topY-1;
   }
                y0=y0b;  pos = CGPointMake( x0 + ( subBlockSize.width), y0);
              GBlock* block=[GBlock newBlockWithPos:pos withSize:subBlockSize parent:tBlock];
                tBlock->figArr[3] = block;     tBlock->figArr[3].hidden = NO;
             tBlock->blockPositionInGameArr[3].x = 4 + 1;
                tBlock->blockPositionInGameArr[3].y = topY;                }
break;
```

GAMEHUB University-Enterprises Cooperation In Game Industry In Ukraine
561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union